Composition de services et supervision : application aux Web Services

Rémi Badonnel

Université Henri Poincaré – Nancy I **École Doctorale IAEM Lorraine**

Composition de services et supervision : application aux Web Services

MÉMOIRE

soutenu le 30 juin 2003

pour l'obtention du

DEA de l'Université Henri Poincaré – Nancy I

(Spécialité Informatique)

par

Rémi BADONNEL

Composition du jury

Membres du jury :	Dominique Méry
	Didier Galmiche
	Noëlle Carbonell
	Olivier Festor
Encadrant :	Laurent Andrey

Laboratoire Lorrain de Recherche en Informatique et ses Applications - Equipe MADYNES

Remerciements

Ayant terminé ce stage de DEA qui s'est déroulé au sein du LORIA, plus précisément dans l'équipe MADYNES, qui travaille sur la supervision des réseaux et des services dynamiques,

je tiens à remercier tout particulièrement :

- Monsieur Olivier Festor, responsable scientifique de l'équipe MADYNES, pour m'avoir accueilli au sein de son équipe et pour son soutien dans la réalisation de ce projet,

- Monsieur Laurent Andrey, mon encadrant universitaire, pour son aide et ses conseils avisés,

- l'ensemble de l'équipe MADYNES, pour l'ambiance conviviale qui règne dans cette équipe,

- le secrétariat du LORIA pour ses renseignements.

Table des matières

Table des figures

Introduction générale

Le développement de l'Internet a favorisé l'échange de données entre partenaires et la multiplication de services en ligne. Des plateformes de services évoluées[1] permettent aujourd'hui la conception, le déploiement et la mise en œuvre de ces services dans des échelles de temps réduites. Au sein de ces plateformes, les services peuvent être perçus comme des composants, et des modèles de composition[2] permettent de les combiner pour aboutir à des services plus élaborés.

Ces services présentent par conséquent une forte dynamicité, d'où la nécessité de créer des plateformes de supervision qui soient adaptées. La richesse des spécifications fournies par les modèles de composition peut constituer un socle pour cette supervision.

Ce stage de DEA consiste à **trouver de nouvelles méthodes permettant d'exploiter les modèles de composition pour faciliter la supervision de services composés**[3].

Pour cela, nous nous sommes placés dans le cadre des Web Services et avons étudié une des fonctions de la supervision : la gestion de la qualité. Nous proposons d'étendre un langage de composition pour qu'il intègre des éléments de qualité de service et nous avons défini une plateforme qui exploite cette extension, et qui permet la gestion de la qualité des Web Services composés par recomposition dynamique.

Dans ce rapport, nous établirons, dans une première partie, un état de l'art des Web Services et de leur supervision. Nous commencerons par présenter le modèle des Web Services : l'architecture générale, les standards utilisés et la composition de services (chapitre 1). Puis, nous détaillerons les différents travaux de recherche en matière de supervision des Web Services (chapitre 2).
Dans une seconde partie, nous présenterons nos contributions : une extension d'un langage de composition pour la qualité de service (chapitre 3), une plateforme pour la gestion de la qualité dans les Web Services composés (chapitre 4) et un prototype de cette plateforme (chapitre 5).

[1] Web Services, CCM (Un glossaire est fourni à la fin du rapport.)
[2] Ces modèles correspondent à des langages qui décrivent des interactions entre services.
[3] Un service composé est un service obtenu par composition.

Première partie

État de l'art

Chapitre 1

Les Web Services

1.1 Introduction

L'objectif de ce chapitre est d'introduire le concept de «Web Services». Nous commencerons par étudier l'architecture générale de cette technologie et le cycle de vie d'un Web Service. Puis, nous présenterons les différents standards sous-jacents : SOAP, WSDL et UDDI. Enfin, nous terminons en présentant le mécanisme de composition de services avec le langage BPEL4WS.

1.2 Définition

Le développement du Web a favorisé le dialogue entre systèmes informatiques et le déploiement d'applications distribuées. Il a également révélé des manques évidents en matière d'intégration et d'interopérabilité entre services. Le modèle des Web Services [14] a été mis en place afin de pallier ces manques. Les Web Services peuvent être définis :

- d'un point de vue *business* : ils décrivent des fonctionnalités exposées par une entreprise sur l'Internet, afin de fournir un moyen d'utiliser ces services à distance. Typiquement, une agence de voyages met à disposition un Web Service pour la réservation de séjours.
- d'un point de vue *technique* : les Web Services correspondent à des applications modulaires, faiblement couplées qui s'exécutent au travers de l'infrastructure Web et qui peuvent être :
  - décrites,
  - publiées auprès d'un annuaire,
  - invoquées par des applications clientes,
  - et orchestrées avec d'autres services.

Les Web Services reposent sur trois standards XML, que nous détaillerons davantage par la suite :

- SOAP, un protocole qui permet d'invoquer à distance les opérations offertes par un Web Service en utilisant des messages XML,
- WSDL, un standard qui permet de décrire l'interface d'un Web Service sous la forme d'un fichier de description en XML,
- UDDI, un protocole d'annuaire qui permet à la fois de publier et de retrouver un Web Service.

1.3 Architecture générale

L'architecture des Web Services repose sur le modèle SOA (*Service-Oriented Architecture*). Ce modèle est né pour répondre aux problèmes d'intégration dans les architectures logicielles, c'est à dire faciliter l'ajout d'un nouveau module dans une application sans remettre en cause toute l'architecture. Aussi, SOA propose un modèle architectural qui permet d'implémenter des systèmes dynamiques dans lesquels l'ajout de nouveaux modules est facile.

L'architecture SOA fait intervenir trois acteurs :

- le *service provider* : il fournit des services et les expose aux utilisateurs en les publiant,
- le *service broker* : il joue le rôle d'annuaire en offrant deux fonctionnalités : la publication et la recherche de services,
- le *service requestor* : il représente un client. Il interroge le *service broker* pour trouver un service, puis invoque ce service en se liant au *service provider*.

1.4 Cycle de vie d'un Web Service

Nous allons présenter l'implémentation de cette architecture dans le cas des Web Services, en prenant l'exemple d'un Web Service d'une agence de voyages pour la réservation de séjours.

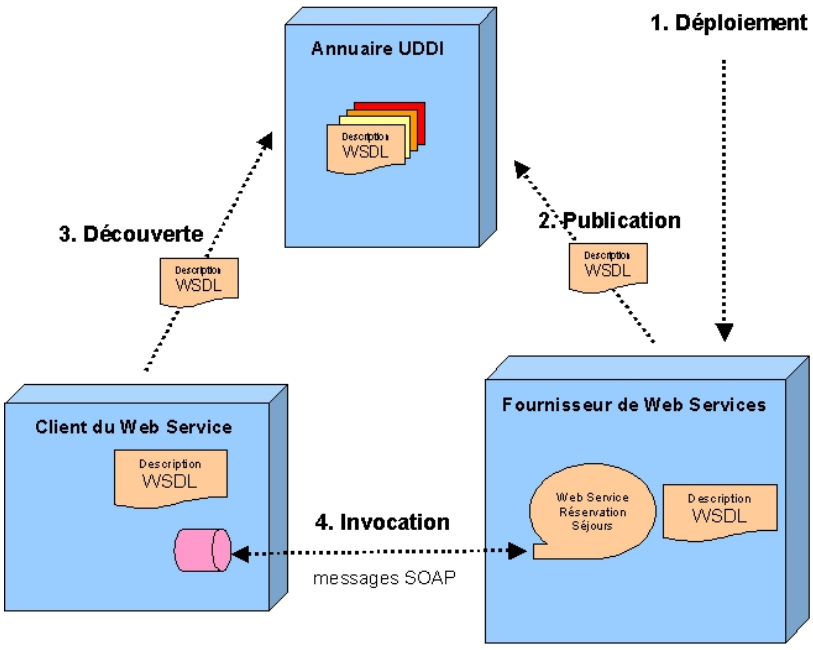


FIG. 1.1 – Cycle de vie d'un Web Service

Le cycle de vie (figure 1.1) se décompose en quatre étapes successives :

- **Déploiement** : le fournisseur de services (l'agence de voyages) déploie le Web Service de réservation sur un serveur et génère une description du service (WSDL). Cette description précise les opérations disponibles et comment les invoquer.
- **Publication** : le fournisseur expose son service de réservations en publiant la description du service auprès d'un annuaire (UDDI).

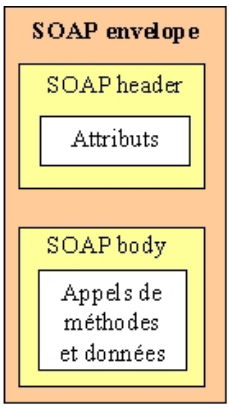
- **Découverte** : le client du Web Service lance une recherche auprès de l'annuaire UDDI pour retrouver la description du service. Cette étape de découverte permet de connaître les différents fournisseurs disponibles et de sélectionner celui qui répond le mieux aux critères du client.
- **Invocation** : Le client utilise la description du service pour établir une connexion avec le fournisseur et invoquer le Web Service (i.e. réserver un séjour).

1.5 Standards sous-jacents

Les trois standards XML sur lesquels reposent les Web Services sont : SOAP, WSDL et UDDI.

1.5.1 SOAP et l'échange de messages

SOAP [10] (*Simple Object Access Protocol*) constitue une pièce maîtresse dans l'architecture des Web Services. Visé par le W3C[4], ce protocole permet la standardisation des messages échangés entre les applications, notamment dans le cadre d'appels de procédures distantes. SOAP s'appuie sur le standard XML et typiquement sur le protocole HTTP, bien qu'il soit ouvert à d'autres modes de transport (SMTP, MOM).



Un message SOAP (figure 1.2) comporte trois parties :

- l'enveloppe (obligatoire) qui contient le nom du message et l'espace de nom (*namespace*),
- l'entête (optionnelle) ou *header* qui est utilisée lorsque le message doit être traité par plusieurs intermédiaires,
- le corps (obligatoire) ou *body* qui contient les opérations qui seront exécutées par le destinataire et les données échangées.

FIG. 1.2 – Message SOAP

1.5.2 WSDL et la description de services

WSDL [11] (*Web Service Description Language*) est un standard XML visé par le W3C, qui est utilisé pour décrire les Web Services et pour permettre aux clients de savoir comment y accéder. Le fichier de description WSDL d'un Web Service contient un ensemble de définitions qui décrit :

- l'**interface du service** c'est à dire les opérations que le service fournit, les formats des données et les protocoles utilisés,
- l'**implémentation du service** c'est à dire l'adresse applicative (URL) pour accéder au service.

[4]World Wide Web Consortium http ://www.w3c.org

1.5.3 UDDI et la découverte de services

UDDI [22] (*Universal Description, Discovery and Integration*) est un standard spécifié par OASIS[5], pour la publication et la découverte des Web Services. UDDI définit à la fois un modèle structurel des données (*UDDI Registry*) et les API que l'opérateur d'annuaire doit fournir pour publier et découvrir les services. Concrètement, un annuaire UDDI permet à un client de récupérer la description WSDL d'un Web Service.

1.6 Composition de services

1.6.1 Principe

La composition de services [8] permet de combiner des Web Services élémentaires afin d'obtenir des services plus élaborés. Typiquement, nous pouvons modéliser le Web Service de réservation de notre agence de voyages comme la composition d'un Web Service de compagnie aérienne, d'un Web Service d'une chaîne d'hôtels, d'un Web Service bancaire et d'un Web Service de livraison.

La composition décrit un ensemble d'interactions ou business process, faisant intervenir différents Web Services. Elle est décrite indépendamment de son implémentation future i.e. elle indique uniquement les types de Web Services nécessaires (Web Service bancaire) mais ne précise pas nominativement les Web Services qui seront utilisés (Web Service de la Banque de France).

Par ailleurs, la composition peut être à plusieurs niveaux en permettant à des Web Services élaborés d'être à leur tour combinés pour construire de nouveaux services.

Elle permet de définir un business process qui est :

- soit **abstrait** : il décrit alors uniquement les interactions observables, i.e. les messages échangés entre Web Services,
- soit **exécutable** : il décrit à la fois les interactions avec l'extérieur et le processus réalisé en interne.

1.6.2 Langage de composition BPEL4WS

Le *Business Process Execution Language* [13] ou BPEL4WS est un langage proposé par IBM, Microsoft et BEA. Il remplace les précédents langages de workflow WSFL et XLANG. BPEL4WS correspond à une grammaire XML qui décrit des business process abstraits ou exécutables. Les business process exécutables peuvent être interprétés et exécutés par un moteur d'orchestration (BPWS4J par exemple).
Comme nous l'avons indiqué précédemment, un business process correspond à une séquence d'opérations ou plus exactement à un flux d'activités [18]. Ces activités peuvent faire intervenir de un à plusieurs Web Services. On distingue les activités de base et les activités structurées.

[5]OASIS est un consortium international d'industriels pour la promotion du commerce électronique http ://www.oasis-open.org

Les **activités de base** permettent :
- d'invoquer une opération d'un Web Service (*invoke*),
- de recevoir une invocation (*receive*),
- de générer une réponse à une invocation (*reply*).

Les **activités structurées** utilisent les activités de base pour décrire :
- des séquences ordonnées (*sequence*),
- des branchements (*switch*, *if*),
- des boucles (*while*),
- des chemins alternatifs (*pick*),
- des exécutions en parallèle (*flow*).

Le langage intègre également un mécanisme de gestion des exceptions (à la manière de java) (*throw*, *catch*), ainsi qu'un mécanisme de compensation (*scope*) qui permet d'annuler une transaction dans son intégralité lorsque celle-ci échoue.
Il constitue une couche supérieure au langage de description WSDL. Il utilise, en effet, WSDL pour définir les opérations de Web Services élémentaires à appeler et pour présenter le business process comme un nouveau Web Service.

1.6.3 Exemple

Reprenons l'exemple d'un Web Service de réservation d'une agence de voyages. Le client invoque le service de réservation en fournissant un message de demande qui contient ses coordonnées (personnels et bancaires) et le séjour choisi. En retour, il obtient un numéro de réservation et recevra sa facture par courrier dans les jours suivants.

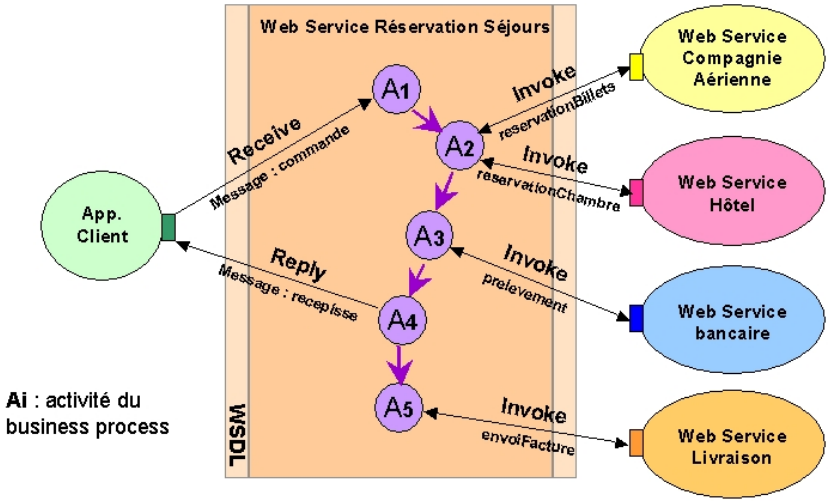


FIG. 1.3 – Modélisation du Web Service de réservation d'une agence de voyages avec BPEL4WS

Ce Web Service de réservation peut être modélisé sous la forme d'un business process (figure 1.3) en combinant différents Web Services élémentaires. Ainsi, à la réception de la demande de réservation, le business process invoque en parallèle le Web Service d'une compagnie aérienne et celui d'une chaîne d'hôtels pour effectuer les réservations. Ensuite, il effectue le prélèvement bancaire

auprès du Web Service d'une banque et donne au client son numéro de réservation. Enfin, il invoque un Web Service de livraison pour l'envoi de la facture papier au domicile du client.

La description BPEL4WS correspondante[6] se décompose en trois parties :

- définition des participants du business process :

```
<partners>
     <partner name="client" partnerRole="clientAgenceDeVoyages">
     <partner name="compagnieAerienne" partnerRole="fournisseurBilletsAvion">
     ...
</partners>
```

- définition des types de données et des messages échangés,

```
<variables>
     <variable name="commande" messageType="typeCommande">
     <variable name="recepisse" messageType="typeRecepisse">
     ...
</variables>
```

- description du business process

```
<sequence>
   <receive partner="client" operation="reservationSejour" variable="commande">
   <flow>
       <invoke partner="serviceCompagnie" operation="reservationBillets">
       <invoke partner="serviceHotel" operation="reservationChambre">
   </flow>
   <invoke partner="serviceBanque" operation="prelevement">
   <reply partner="client" operation="reservationSejour" variable="recepisse">
   <invoke partner="serviceLivraison" operation="envoiFacture">
</sequence>
```

1.7 Synthèse

Les Web Services améliorent l'intégration et l'interopérabilité entre services à travers l'infrastructure Web en utilisant différents standards XML : SOAP pour l'échange de messages, WSDL pour la description de services et UDDI pour la publication et la découverte de services. Ils reposent sur une architecture orientée service (SOA) et correspondent à des composants logiciels qui peuvent être combinés, grâce à un langage de composition, pour former de nouveaux services plus élaborés.

La forte dynamicité et la nature distribuée de ces services nécessitent de créer des infrastructures de supervision adaptées.

[6] Cette description a été volontairement simplifiée dans cet exemple.

Chapitre 2

La supervision des Web Services

2.1 Introduction

La supervision d'un service consiste à le surveiller et à agir sur ses paramètres opérationnels pour qu'il satisfasse les demandes des utilisateurs et les contraintes des fournisseurs. Les Web Services se distinguent des services classiques par leur forte dynamicité et nécessitent de créer de nouvelles infrastructures de supervision mieux adaptées.

Dans ce chapitre, nous analyserons les différentes fonctions assurées par la supervision dans le cadre des Web Services, pour nous permettre par la suite de mieux positionner notre projet. Nous insisterons sur deux architectures importantes : le réseau *WSMN* pour le monitorage de services composés et la plateforme *WSLA* pour la gestion de contrats de service.

2.2 Monitorage et gestion des fautes

Les Web Services étant déployés sur le réseau, il est important de connaître précisément l'état d'un Web Service. En particulier, le monitorage consiste à évaluer la disponibilité d'un Web Service par le biais de mesures.

Cette évaluation ne doit pas se restreindre à spécifier si le service fonctionne ou non. Elle doit comprendre un ensemble d'indicateurs significatifs, qui portent sur les performances du service, tels que le nombre de messages SOAP échangés, le nombre de transactions réussies ou le temps de traitement moyen d'une requête.

Lorsque le fonctionnement anormal d'un Web Service est détecté, des messages d'alarme sont émis (gestion de fautes).

2.2.1 Utilisation de proxys

Le monitorage des Web Services repose sur l'analyse des messages SOAP qui sont échangés entre le Web Service et ses clients. C'est pourquoi, un modèle fréquent dans la supervision des Web Services est l'usage de proxys (figure 2.1), qui permettent l'instrumentation du service.

Un exemple d'implémentation de ce modèle est l'utilisation de la plateforme SOAP Axis comme proxy et l'instrumentation de celle-ci à l'aide d'un agent JMX [16]. L'agent enregistre

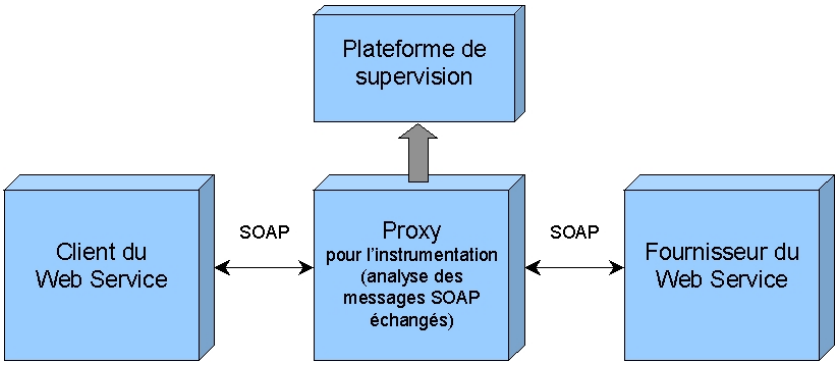


FIG. 2.1 – Utilisation d'un proxy pour l'instrumentation

l'ensemble des messages SOAP en entrée et sortie d'Axis et transmet les mesures auprès d'une plateforme de supervision.

2.2.2 Monitorage des Web Services composés

Les services composés sont, dans le cas général, distribués. L'architecture *Web Services Management Network* ou WSMN [23] a été proposée par les laboratoires HP pour faciliter leur monitorage.

WSMN est un réseau d'overlay, au-dessus de SOAP, constituait d'un ensemble d'intermédiaires qui communiquent entre eux (figure 2.2).

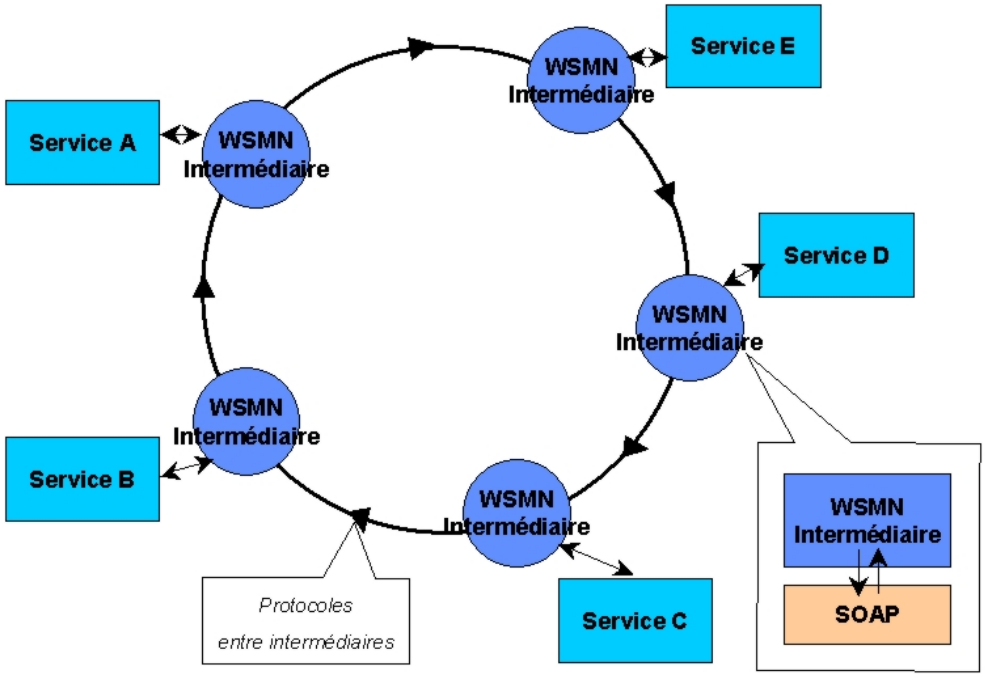


FIG. 2.2 – Architecture du réseau WSMN

Chaque intermédiaire joue le rôle de proxy entre deux Web Services de la composition. Il permet le monitorage mais aussi la corrélation de messages (détection des dépendances entre Web Services) et la gestion de fautes.

Les échanges entre intermédiaires sont réalisés à l'aide de trois protocoles :

- *Life-Cycle Protocol* : il permet la construction du réseau à partir d'un ensemble d'intermédiaires indépendants,
- *Measurement Protocol* : les mesures effectuées sur les messages doivent parfois être réalisées sur plusieurs sites différents. Ce protocole permet l'échange de mesures entre intermédiaires.
- *Assurance Protocol* : ce protocole permet d'émettre des messages d'alarme pour avertir les intermédiaires en cas de détection de faute.

Les intermédiaires ont besoin d'un modèle qui représente le Web Service composé, afin de pouvoir attacher à ce modèle les mesures effectuées :

- Si le business process (BPEL4WS) est connu, un modèle du service est automatiquement généré dans une base de données (*model repository*).
- Sinon, les intermédiaires disposent d'un module de corrélation [6], qui en ajoutant des éléments aux entêtes des messages SOAP, permettent d'analyser les dépendances et de générer un modèle du Web Service.

Le réseau WSMN constitue une approche importante dans le monitorage et la gestion des fautes des Web Services composés.

2.3 Gestion de la configuration

Un autre rôle joué par la supervision consiste à configurer les Web Services. Pour se faire, des agents sont déployés sur les plateformes de services. Ils permettent de représenter les Web Services sous la forme d'objets dont on peut paramétrer les attributs.
Les travaux en la matière restent encore à développer. Jusqu'à présent, la gestion de la configuration des Web Services se résume à activer ou désactiver un Web Service (en instrumentant la plateforme SOAP).

2.4 Gestion d'instances multiples et grid computing

Un Web Service est amené à satisfaire plusieurs clients simultanément, ce qui est permis en autorisant plusieurs instances d'un même service de coexister. Ces instances peuvent être traitées sur plusieurs machines différentes (figure 2.3), à travers une architecture de grid computing [19].

Le système de supervision doit permettre de gérer la charge d'un Web Service à travers des mécanismes de files d'attente, il doit s'assurer que le Web Service satisfait le maximum de la clientèle en un mininum de temps.

Par ailleurs, le système peut intégrer un mécanisme de classes pour la clientèle. Ceci permet de privilégier les clients qui ont souscrit pour un niveau de service supérieur.

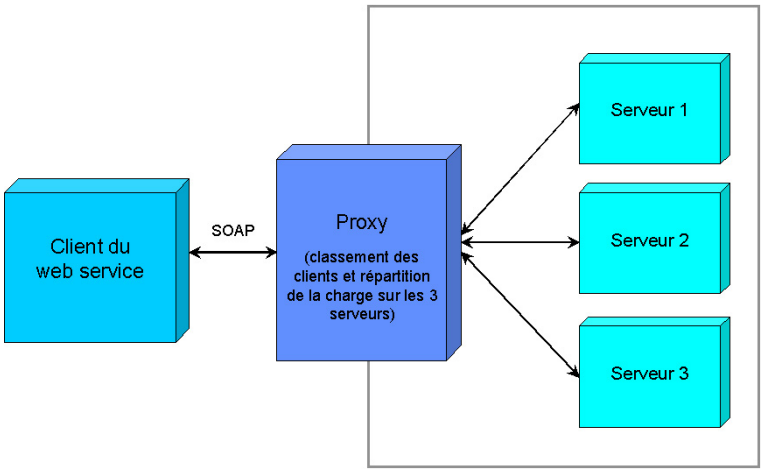


FIG. 2.3 – Web Services et grid computing

2.5 Gestion de la qualité

La gestion de la qualité joue un rôle essentiel dans la supervision. Elle consiste à assurer un niveau de service donné (performance, disponibilité) pour un Web Service.

2.5.1 Contrat de services

La gestion de la qualité passe souvent par l'établissement d'un contrat de services (*Service Level Agreement*) entre un client et un fournisseur.

Ce contrat [1] spécifie les objectifs à respecter en terme de qualité de service. Au cours de l'utilisation du service, des mesures sont réalisées par l'infrastructure de supervision afin de vérifier que le contrat est bien respecté [12]. Le cas échéant, des pénalités et des mesures correctives doivent être prises.

L'ouvrage *Foundations of Service Level Management* [20] décrit les différents éléments d'un contrat de services :

- *Parties* : les entités concernées par le contrat i.e. les signataires (typiquement le fournisseur et le client) et éventuellement des entités tierces pour effectuer des mesures,
- *Term* : la durée de validité du contrat,
- *Service level objectives* : les objectifs à respecter (en matière de disponibilité, performance, sécurité),
- *Service level indicators* : les métriques (temps de réponse par exemple) que l'on utilise pour vérifier que les objectifs sont atteints,
- *Penalties* : les sanctions appliquées en cas de non respect du contrat,
- *Scope, limitations* : le cadre exact du contrat (portée, limitation, exclusion).

Le contrat de services permet de confronter les besoins du client avec les limites du fournisseur. Il doit être à la fois flexible pour s'adapter au plus grand nombre de cas et suffisamment contraint pour qu'il n'y ait pas de malentendus sur la qualité du service fourni.

2.5.2 Langages de contrat de services pour les Web Services

Deux langages *WSLA* [3] et *Web Service SLA* [5] ont été proposés afin de formaliser ces contrats dans le cas des Web Services.

Ils utilisent des syntaxes différentes. Néanmoins, nous retrouvons, dans chacun des cas, les sections caractéristiques d'un contrat de services, comme le confirme le tableau comparatif.

	WSLA	**Web Service SLA**
Proposé par	IBM Research	HP Labs
Parties	**Parties** Signatory Parties Supporting Parties	**Partner**
Service Level Objectives	**Service Description** Service Operations SLA Parameters	**SLO** Clause Measured Item
Service Level Indicators	Metrics Functions Schedule	Item evalFunc (agrégation) evalWhen/evalOn
Evaluation	**Obligations** Predicat	 evalFunc (évaluation)
Penalties	Action	evalAction
Term	Validity Period	Date Constraint & Date Time

FIG. 2.4 – Tableau comparatif WSLA et Web Service SLA

Ces langages définissent des métriques de haut niveau *SLA Parameters* ou *MeasuredItem* (temps de réponse, débit), qui sont exprimées sous la forme de métriques de bas niveau *Metrics* ou *Item* (compteurs, gauges). Á partir de ces langages sont construits différentes architectures de supervision, qui permettent d'assurer le respect des contrats de service.

2.5.3 Plateforme WSLA

Le *Web Service Level Agreement (WSLA) framework* [3] est une infrastructure proposée par IBM Research et qui permet de définir et de monitorer des contrats de services dans le cadre des Web Services. Elle comprend le langage WSLA (dont nous avons parlé précédemment) flexible et extensible basé sur XML pour exprimer les contrats et une architecture composée de différents services pour le monitorage des contrats.

Les objectifs de ce framework sont multiples :

- fournir un langage formel de contrats de service pouvant s'adapter à des cas multiples,
- faciliter l'intégration de systèmes dans le domaine du commerce électronique, en automatisant le processus de négociation et de déploiement de contrats de service,
- permettre la délégation des tâches de supervision à des entités externes,
- configurer automatiquement les ressources en fonction des contrats établis.

Pour répondre à ces objectifs, l'architecture WSLA (figure 2.5) se décompose en différents services élémentaires (services de négociation, de déploiement, de mesure, d'évaluation et de management) qui permettent de gérer les contrats de services tout au long de leur cycle de vie.

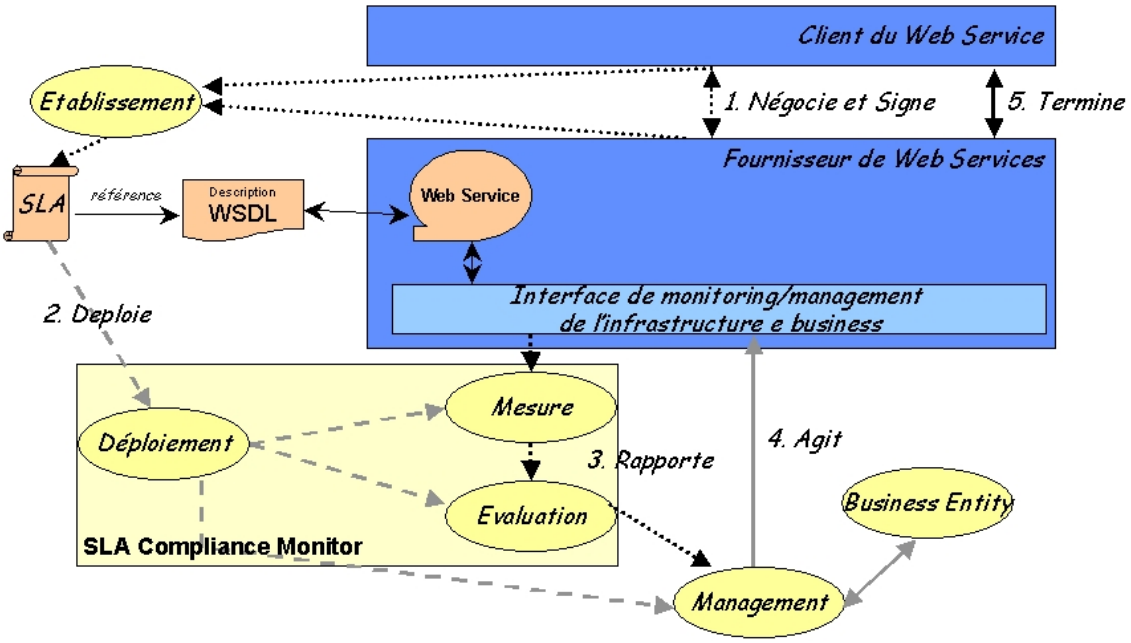


FIG. 2.5 – Architecture du framework WSLA et cycle de vie d'un contrat de services

Détaillons le rôle de chacun de ces services en décrivant le cycle de vie d'un contrat de services :

- **Etape 1 : Négociation et établissement d'un contrat**
  Cette opération est réalisée par le service d'établissement de contrats. Le contrat de services doit être négocié et signé par le client et le fournisseur du Web Service. La difficulté de cette étape est de confronter les besoins du client avec la qualité de service que peut offrir le fournisseur. C'est pourquoi, des extensions à la description WSDL, appelées WSEL (Web Services Endpoint Language) et WSOL [24] (Web Service Offerings Language), ont été proposées pour compléter la description d'un Web Service avec des éléments non fonctionnels portant sur la qualité et la tarification du service.
- **Etape 2 : Déploiement du contrat**
  Le service de déploiement est chargé de vérifier la validité du contrat de services et de répartir les différents éléments qu'il contient aux services concernés.
- **Etape 3 : Mesure de qualité de service et reporting**
  Cette étape fait intervenir le service de mesure et le service d'évaluation. Ces deux services peuvent être externalisés et placés sous le contrôle d'une entité tierce. Le service de mesure récupère les mesures et les agrège pour calculer la valeur des paramètres du contrat. Le service d'évaluation compare les valeurs obtenues par rapport au contrat et émet des alarmes en cas de violation.
- **Etape 4 : Management et actions correctives**
  En fonction des alarmes émises, le service de management applique des actions correctives telles que spécifiées dans le contrat de services. Avant d'appliquer ces actions, il demande une autorisation auprès d'un administrateur.
- **Etape 5 : Terminaison**
  Le contrat de services prend fin lorsqu'une clause du contrat a été violée ou lorsque la date de validité a été dépassée.

Les services de mesure, d'évaluation et de déploiement ont été implémentés en tant que Web Services à l'aide du toolkit WSTK[7] d'IBM. Ils forment un outil *SLA compliance monitor*, qui permet de vérifier le respect d'un contrat de services.

2.6 Gestion du changement

Comme nous l'avons précisé précédemment, les Web Services présentent une grande dynamicité : les opérations qu'ils offrent évoluent considérablement dans le temps. Par conséquent, la supervision assure des fonctions de *versioning*, qui permettent d'améliorer la transition d'une version d'un Web Service à une autre, en limitant les problèmes d'incompatibilité qui pourraient survenir.

2.7 Gestion de la sécurité

Des mécanismes de sécurité doivent être implémentés pour assurer la confidentialité et l'intégrité des données. La supervision gère l'authentification des clients et leur droit d'accès, elle comprend également des mécanismes pour limiter les attaques pirates telles que le dénis de service.

2.8 Synthèse et positionnement

La supervision des Web Services couvre différentes fonctionnalités, du monitorage d'un service à sa configuration en passant par la gestion de la qualité et celle de la sécurité.
L'utilisation de proxys est un modèle fréquent dans la supervision : ils permettent d'instrumenter les Web Services (analyse des messages SOAP), mais peuvent également servir à répartir la charge lorsque le Web Service est déployé sur plusieurs serveurs.
Le *Web Service Management Network* (WSMN) et le *WSLA framework* sont deux infrastructures importantes dans la supervision des Web Services. WSMN est un réseau d'overlay qui utilise un ensemble de proxys pour le monitorage des services composés. La plateforme WSLA permet, quant à elle, le management de contrats de service.

Les recherches menées en matière de supervision des Web Services n'exploitent pas ou très peu les modèles de composition[8]. Ceux-ci fournissent pourtant des spécifications non négligeables sur les Web Services composés.
Notre projet constitue une nouvelle approche permettant d'exploiter ces modèles pour la supervision des Web Services. Nous nous positionnons dans le cadre de la gestion de la qualité pour les Web Services composés.

[7] Le toolkit WSTK est une plateforme pour le développement et le déploiement de Web Services.
[8] langage de composition BPEL4WS par exemple

Deuxième partie

Contributions

Démarche

Les Web Services sont caractérisés par une forte dynamicité et par le fait qu'ils peuvent être composés. Par conséquent, les plateformes de supervision doivent être adaptées à cette nature. La richesse des spécifications fournies par les modèles de composition[9] peut faciliter cette supervision.

En particulier, notre travail de recherche porte sur la gestion de la qualité dans les Web Services composés et consiste à exploiter un modèle de composition pour permettre cette gestion.

Pour se faire, nous proposons d'étendre un langage de composition (expression d'un business process) pour qu'il intègre des contraintes de qualité. Cette extension facilitera l'exploitation du modèle de composition pour la gestion de la qualité. Dans notre cas, elle sera utilisée par une plateforme de supervision permettant la gestion de la qualité dans les Web Services par recomposition dynamique.

Dans ce contexte, nous commencerons cette seconde partie en présentant une extension au langage BPEL4WS pour la qualité de service. Puis, nous détaillerons notre plateforme pour la gestion de la qualité dans les Web Services composés par recomposition dynamique ; cette recomposition étant dictée par les contraintes de qualité de notre extension. Enfin, nous présenterons un prototype de cette plateforme.

[9] langages de composition

Chapitre 3

Extension d'un langage de composition pour la qualité de service

3.1 Introduction

Dans ce chapitre, nous proposons une extension au langage de composition BPEL4WS pour qu'il intègre des contraintes de qualité. Les langages de composition permettent de définir un business process entre différents Web Services pour aboutir à un service plus élaboré (composition de services). Il est important de pouvoir spécifier en amont, c'est à dire, dès l'étape de conception, des contraintes de qualité portant sur la composition. Ces contraintes n'ont pas un caractère obligatoire, mais permettent de définir les limites de fonctionnement d'un Web Service composé et peuvent consistuer un support pour sa recomposition dynamique.

3.2 Principe

L'objectif de cette extension est d'exprimer des contraintes de qualité de service, ayant une portée donnée, dans une composition de services.

En effet, définir la qualité globale d'un Web Service composé n'est pas satisfaisant pour le fournisseur du Web Service. Il faut pouvoir spécifier la qualité de service au sein même du business process en fonction des activités qui sont réalisées. Ainsi, dans l'exemple du Web Service de réservation, l'activité correspondant à l'envoi de la facture pourra porter une contrainte sur le temps d'exécution (temps de livraison) tandis que l'activité correspondant au prélèvement bancaire pourra porter une contrainte sur le coût (frais bancaires).

Les contraintes de qualité de services peuvent être classifiées selon leur portée : **contraintes pour une activité de base**, **pour une activité structurée** ou **entre activités**.

3.2.1 Contraintes de qualité pour une activité de base

Les activités de base d'un business process permettent de recevoir une invocation, d'émettre une réponse ou d'invoquer une opération d'un Web Service. Les deux premières activités portent sur les points d'entrée du business process. La dernière activité qui correspond à l'invocation d'un Web Service peut être fortement contrainte par des critères de qualité.

Ces contraintes (figure 3.1) portent sur les caractéristiques du Web Service invoqué : disponibilité, performance, niveau de charge, sécurité ou coût. Elles déterminent à la fois la métrique utilisée et les limites à respecter.

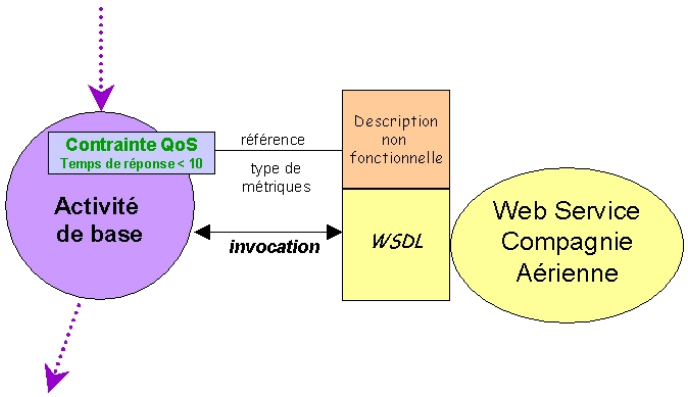


FIG. 3.1 – Contraintes de qualité pour une activité de base

Les travaux portant sur la description non fonctionnelle de Web Services (WSLA offering, WSEL, WSOL) doivent être mis en parallèle à notre démarche. Ces travaux permettent de définir clairement l'offre faite par les fournisseurs. Nous proposons de pouvoir réutiliser dans ces fichiers de description la partie qui spécifie les métriques. De cette façon, il est possible de définir des types de métriques, tout comme il existe des types d'opérations dans la description WSDL des Web Services.

3.2.2 Contraintes de qualité pour une activité structurée

Une activité structurée correspond à un ensemble d'activités de base interconnectées. À l'étape de conception, il est souvent plus facile de spécifier le niveau de service à l'échelle d'une activité structurée. Les principales contraintes (figure 3.2) de qualité qui peuvent être utilisées sont :

- la durée de l'activité (temps de traitement),
- le temps de réponse,
- le débit (nombre d'instances de l'activité par minute),
- et le coût financier.

La qualité d'une activité dépend directement des activités de base utilisées et de la façon dont elles sont assemblées (séquence, parallélisme). Les activités structurées peuvent à leur tour être combinées pour former une nouvelle activité structurée, celle de plus haut niveau correspondant au business process lui-même. Par conséquent, en spécifiant la QoS pour une activité structurée, nous nous attachons à une vision ensembliste d'un business process.

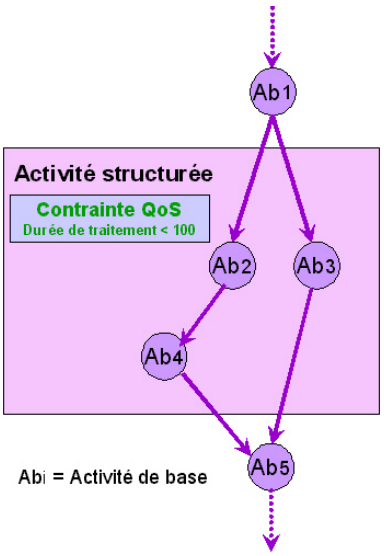


FIG. 3.2 – Contraintes de qualité pour une activité structurée

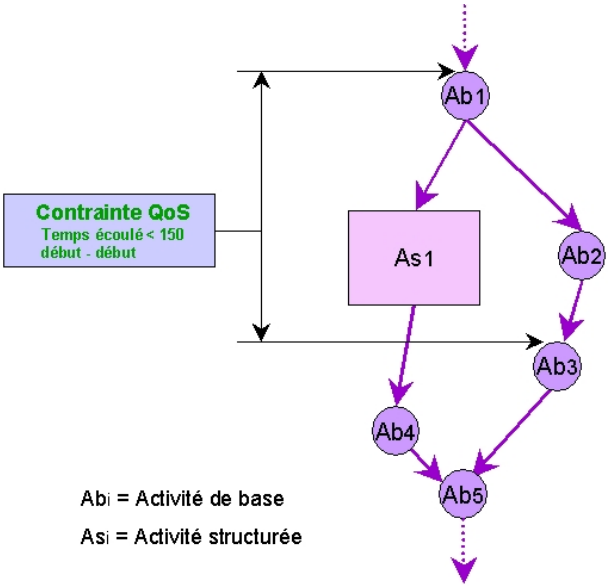


FIG. 3.3 – Contraintes de qualité entre activités

3.2.3 Contraintes de qualité entre activités

Cette vision ensembliste n'est pas suffisante pour offrir un large champ dans l'expression des critères de qualité. Il doit être également possible de spécifier des contraintes entre deux activités (de base ou structurée).

Dans le cas du service de réservation de séjours modélisé par un business process, nous devons pouvoir spécifier une durée limite entre le prélèvement du prix du séjour et l'envoi de la facture au client (voire la réception de celle-ci par le client).

Ces contraintes (figure 3.3) peuvent porter sur des activités appartenant à des ensembles distincts (deux activités de base appartenant à deux activités structurées différentes). Les principaux critères de qualité sont le temps écoulé et le débit entre deux activités.

3.2.4 Cohérence

Une des étapes les plus complexes dans la spécification de la qualité de service d'un business process est de s'assurer de la cohérence entre les différents critères exprimés. Dans le cas des contraintes portant sur les activités de base et sur les activités structurées, il est facile de détecter des incohérences en établissant des règles sur des ensembles. Ainsi, la durée d'une activité structurée de type séquence ne peut pas être inférieure à la somme des durées des activités de base qui la composent. La vérification de la conformité d'un business process constitue un travail à part entière.

3.3 Extension du langage BPEL4WS

Nous proposons d'étendre le langage BPEL4WS avec un nouveau élément : QoSConstraint qui permet de définir des critères de qualité dans un business process.

3.3.1 Définition des contraintes de qualité

Les contraintes de qualité sont définies dans une section définition au début d'un document BPEL4WS. Chaque contrainte possède les attributs :

- name : le nom de la contrainte de qualité,
- type : le type de contrainte c'est à dire pour une activité de base (basic), pour une activité structurée(structured) ou entre activités (inter),
- metric : la métrique utilisée (temps de réponse, débit),
- predicat : la contrainte sur la métrique (inférieure à 100).

```
< QoSConstraint name="contrainteA" type="structured"
                                   metric="executionTime" predicat="<200" >

< QoSConstraint name="contrainteB" type="basic"
                                   metric="cost" predicat="<500" >
```

3.3.2 Portée des contraintes de qualité

Une fois définies, les contraintes de qualité doivent être attachées au business process. Pour se faire, nous proposons de faire référence au nom de la contrainte au moment de la description d'une activité (de base ou structurée). Ces références seront considérées comme uniques excepté dans le cas d'une contrainte entre activités. Dans ce cas, la référence au nom s'établit exactement deux fois : ceci afin de pouvoir préciser les deux activités entre lesquelles doit s'établir la contrainte.

```
<sequence QoSConstraints="contrainteA">
   <receive partner="client" operation="reservationSejour" variable="commande">
   <flow>
       <invoke partner="serviceCompagnie" operation="reservationBillets">
       <invoke partner="serviceHotel" QoSConstraints="contrainteB"
               operation="reservationChambre">
   </flow>
   <invoke partner="serviceBanque" operation="prelevement">
   <reply partner="client" operation="reservationSejour" variable="recepisse">
   <invoke partner="serviceLivraison" operation="envoiFacture">
</sequence>
```

3.3.3 Référence à des types de métriques

Dans le cas d'une activité de base (invocation d'un Web Service élémentaire), il est possible de spécifier la métrique utilisée en réutilisant les fichiers de description non fonctionnelle du Web Service (WSOL, WSEL, WSLA offering). Un attribut supplémentaire nommé metricType est alors employé pour préciser l'URL du fichier de description.

```
< QoSConstraint name="contrainteC" type="basic"
                                   metricType="www.webservice.com/prelevement.wsol"
                                   metric="responseTime" predicat="<10" >
```

3.3.4 Statut et classement

Les contraintes de qualité d'un document BPEL4WS pourraient être spécifiées uniquement à titre indicatif. Une balise *status* qui prendrait les valeurs *recommended* ou *required* pourrait être ajoutée à la définition d'une contrainte. Par ailleurs, nous pourrions établir un classement ou des règles de priorité entre les contraintes à l'aide d'un attribut rank.

```
< QoSConstraint name="contrainteA" status="required" ... >
< QoSConstraint name="contrainteB" status="recommended" rank="1" ... >
< QoSConstraint name="contrainteC" status="recommended" rank="2" ... >
```

3.4 Travaux annexes

Cette approche ouvre de nouvelles perspectives dans l'élaboration des business process. Des travaux complémentaires pourraient être réalisés. D'une part, des analyseurs doivent être employés pour vérifier la conformité d'un document BPEL4WS avec qualité de service, l'étape la plus complexe consistant à détecter les incohérences entre contraintes. D'autre part, ces contraintes peuvent servir d'indicateurs pour établir les contrats de service et pour évaluer la qualité de service d'un service composé et plus généralement pour la gestion de la qualité.

3.5 Synthèse

Les langages de composition de service peuvent être étendus afin d'exprimer des contraintes de qualité de service. Ces contraintes reposent sur des métriques (disponibilité, performance, sécurité) et ont une portée donnée (activité de base, activité structurée ou entre activités). Nous proposons ici une extension pour le langage BPEL4WS qui repose sur la définition de contraintes de qualité QoSConstraint qui sont ensuite référencées par les activités du business process.

Cette extension permettra d'exploiter les langages de composition pour la gestion de la qualité dans les Web Services composés.

Chapitre 4

Plateforme de gestion de la qualité par recomposition dynamique pour les Web Services composés

4.1 Introduction

Nous proposons dans ce chapitre une plateforme pour la gestion de la qualité dans les Web Services composés. Elle permet la recomposition dynamique d'une composition de services par routage applicatif. L'objectif de cette structure est de resélectionner les partenaires d'une composition afin de respecter les contraintes de qualité exprimées par le langage de composition BPEL4WS muni de notre extension.
Cette recomposition peut s'opérer soit parce que les partenaires courants ne permettent plus de respecter les contraintes de qualité, soit parce que de nouveaux partenaires plus performants sont apparus.

4.2 Plateforme de gestion de la qualité par recomposition

4.2.1 Principe

Une composition est décrite sous la forme d'un business process. Ce dernier orchestre un ensemble de partenaires dans le but d'obtenir une tâche donnée. Notre plateforme permet de sélectionner, durant la durée de vie d'un Web Service composé, les partenaires les plus efficaces pour respecter un niveau de qualité de service. Pour se faire, elle analyse les contraintes de qualité du business process qui sont exprimées avec BPEL4WS (muni de son extension). Puis, elle établit un classement des partenaires qui correspondent au mieux et compose dynamiquement le Web Service composé. Si la qualité de service se dégrade ou si de nouveaux partenaires apparaissent, la plateforme modifie le classement et recompose le service.

4.2.2 Architecture générale

La plateforme est hébergée par le partenaire responsable de la composition. Elle interagit avec un annuaire UDDI privé qui utilise une extension pour la description non fonctionnelle[10] et utilise le Web Service Management Network pour le monitorage du service composé.

[10]Cette extension (WSEL,WSOL) fournit des indications sur la qualité offerte et le coût d'un Web Service.

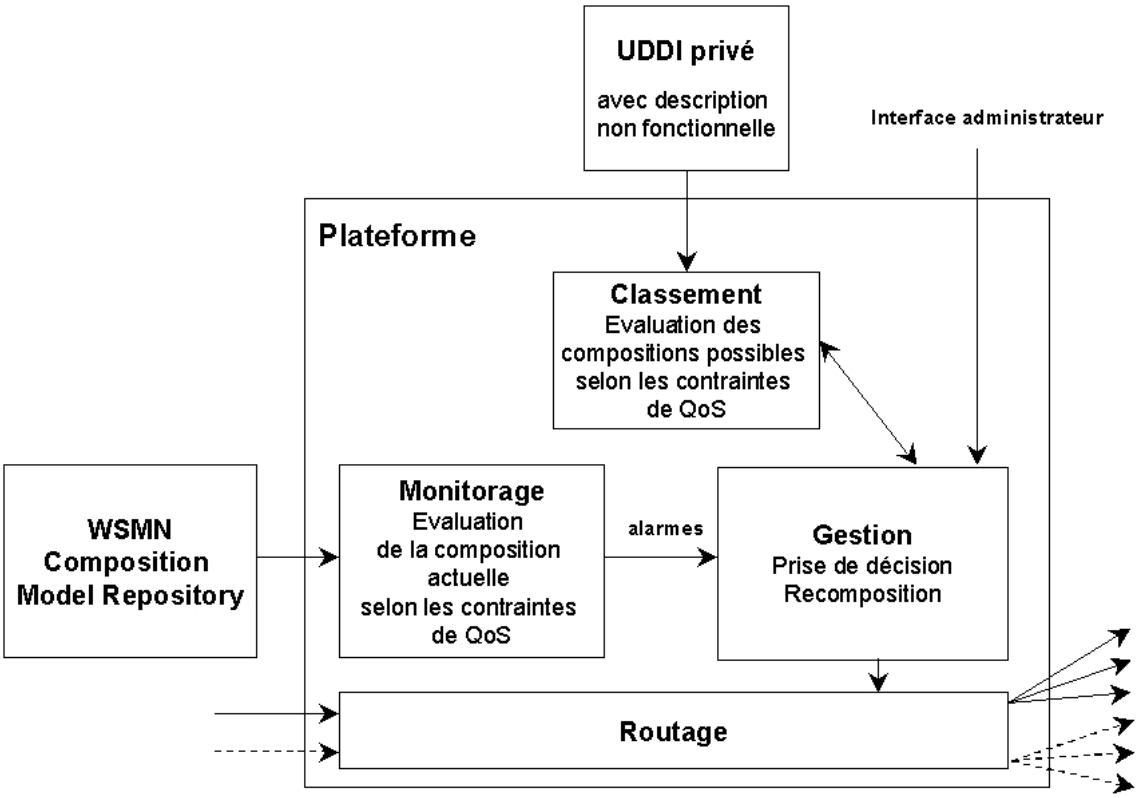


FIG. 4.1 – Architecture générale de la plateforme de gestion de la qualité

Cette plateforme (figure 4.1) peut être décomposée en 4 systèmes :

- **Système de classement**
  Ce système est chargé d'évaluer quelles sont les compositions potentielles de partenaires qui permettent de respecter les contraintes de qualité du business process. Pour se faire, il interroge l'annuaire privé pour connaître le niveau de service offert par chaque fournisseur, puis utilise des règles de déduction afin d'évaluer si les compositions conviennent. Lorsque de nouveaux fournisseurs apparaissent, le classement est automatiquement mis à jour.
- **Système de monitorage**
  Il permet d'évaluer les contraintes de qualité définies dans le business process, pour la composition de services courante. Ce système utilise les données fournies par le *model repository*[11] du *Web Service Management Network* et des règles de déduction afin d'évaluer le respect des critères. En cas de non respect, ce système émet des alarmes auprès du système de gestion.
- **Système de gestion**
  Il détermine quelle est la composition courante. Lorsqu'il reçoit les alarmes émises par le système de monitorage ou des consignes de l'interface administrateur, il interroge le système de classement pour sélectionner la composition la plus efficace. Dès que le choix a été effectué, il appelle le système de routage pour modifier la composition.
- **Système de routage**
  Le système de routage met en correspondance le moteur de workflow du business process avec les partenaires définis par le système de gestion.

[11] base de données contenant les mesures effectuées par WSMN sur le business process

4.3 Fonctionnement de la plateforme

La plateforme utilise les contraintes de qualité du business process et recompose le Web Service afin d'assurer un niveau de qualité. Son fonctionnement peut se décrire en cinq étapes.

4.3.1 Recherche de nouveaux services

La plateforme utilise un annuaire UDDI privé pour la recherche de nouveaux services. Cet annuaire utilise une extension pour permettre une description non fonctionnelle des services disponibles. Il est mis à jour par l'entité responsable du business process. Des mises à jour automatiques à partir d'annuaires publics sont envisageables mais toujours après validation d'un responsable. L'extension fournit des informations sur la qualité de service et sur la qualité de l'expérience (historique et évaluation personnel du prestataire du Web Service composé).
L'usage d'un annuaire privé se justifie par la nécessité d'avoir des informations validées et fiables qui gèrent le retour d'expérience en intègrant des critères de fidélité.

4.3.2 Classement des partenaires par critères

Les informations fournies par l'annuaire permettent d'établir un classement des compositions potentielles selon les contraintes de qualité. Cette étape est réalisée par le système de classement (figure 4.2). Celui-ci analyse les contraintes de qualité et les activités du business process et utilise des règles de déduction pour sélectionner les meilleurs partenaires.

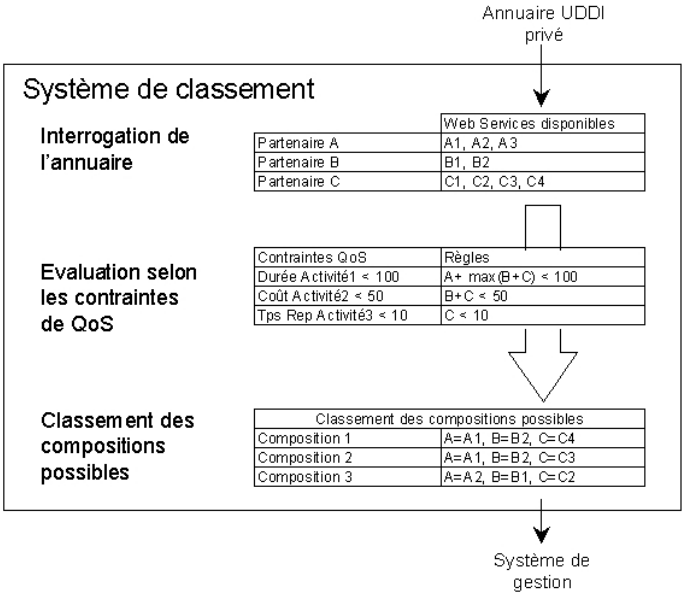


FIG. 4.2 – Système de classement

Etablir des règles de déduction représente un travail conséquent dans les cas les plus complexes, ces règles peuvent s'inspirer des travaux sur la qualité de service dans les workflows [15]. Cependant, ces règles peuvent être facilement trouvées dans les cas les plus simples. Un exemple de règles : une activité structurée S correspond à l'invocation en parallèle d'un service A et d'un service B. Si une contrainte de qualité impose que l'activité structurée S dure au plus 30s alors

chacune des invocations aux deux services doivent avoir une durée inférieure à 30s.

La sélection de la meilleure composition peut nécessiter l'usage de la recherche opérationnelle, si le nombre de contraintes du business process est élevé. Cependant, le temps de traitement pour établir les classements est borné dans la mesure où la sélection repose sur un annuaire privé.

Par ailleurs, ce classement doit intégrer des critères de fidélité pour éviter les mécanismes d'oscillation entre plusieurs compositions potentielles. Ces critères peuvent correspondre à la fréquence avec laquelle la composition (ou plutôt son implémentation) a été utilisée et le nombre de transactions qui ont abouti.

4.3.3 Monitorage de la composition courante

Afin d'évaluer la composition courante, le système de monitorage (figure 4.3) utilise les même règles de déduction mais cette fois-ci à partir des données du monitorage. Nous proposons d'utiliser le *model repository* généré par le *Web Service Management Network* d'Akhil Sahai. Ce *model repository* est une représentation du business process, pour laquelle chacune des branches est complétée par des mesures de qualité de service. Les règles de déduction permettent d'évaluer si les contraintes de qualité sont respectées. Si une des contraintes n'est pas respectée, le système de monitorage émet une alarme.

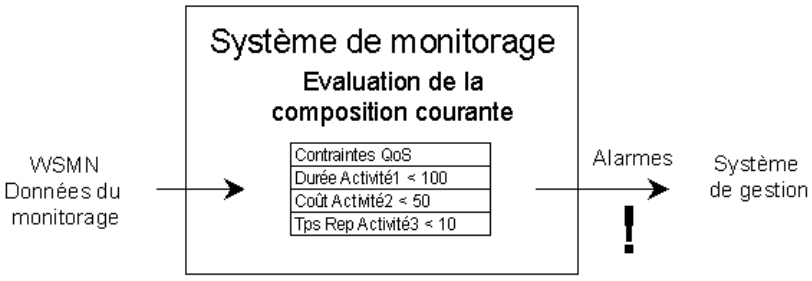


FIG. 4.3 – Système de monitorage

4.3.4 Gestion de la recomposition

La recomposition est gérée par le système de gestion (figure 4.4). Celui-ci écoute les alarmes du système de monitorage ainsi que les consignes provenant de l'interface administrateur. À partir de ces informations et en utilisant les données fournies par le système de classement, il évalue si une recomposition est nécessaire : il compare s'il est plus intéressant de changer de configuration ou de rester dans la configuration actuelle (au risque de dégrader la QoS).

L'étape de gestion est cruciale et ne peut s'opérer que dans une fenêtre spatiale donnée, afin de permettre la terminaison des instances en cours du business process. Dans l'exemple d'une agence de voyage, il n'est pas possible de changer de banque alors qu'une opération bancaire est en cours.
Dès qu'une décision est prise, le système de gestion émet des ordres auprès du système de routage.

4.3.5 Routage

Le routage permet de faire correspondre à une opération donnée du business process, le bon fournisseur (c'est à dire celui choisi pour la composition courante). Le système de routage

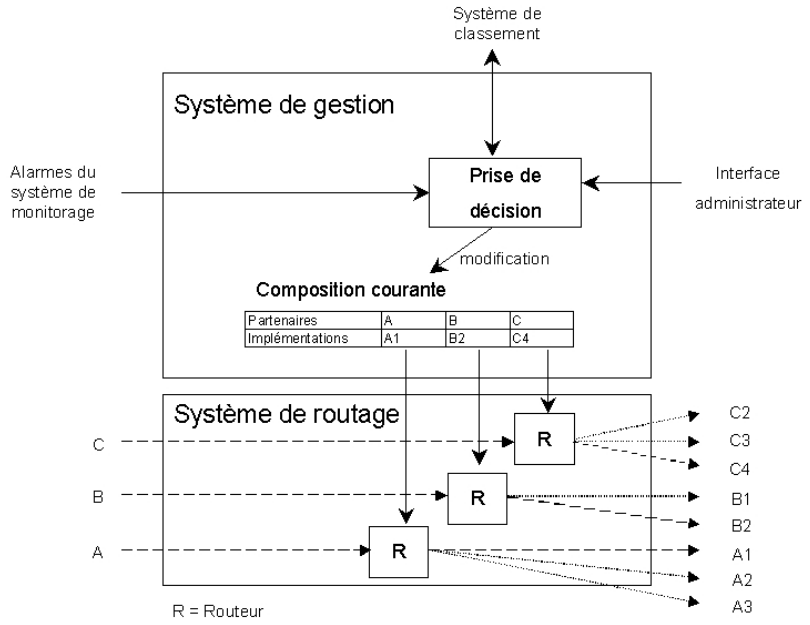


FIG. 4.4 – Systèmes de gestion et de routage

(figure 4.4) gère l'ensemble des opérations à effectuer. Il joue le rôle d'intermédiaire et utilise par conséquent les même interfaces en entrée que les interfaces des fournisseurs auxquels il fait appel.

4.4 Travaux futurs

Des travaux futurs doivent être réalisés en matière de certification, vis à vis des critères de QoS affichés par les fournisseurs. D'autre part, les systèmes de gestion et de routage devraient être étendus pour pouvoir utiliser plusieurs Web Services simultanément pour réaliser une même opération (en s'inspirant des travaux sur le grid computing [19]). Par exemple, le partenaire A pourrait être représenté par A1 à 60% et A2 à 40%.

4.5 Synthèse

Nous proposons une plateforme pour la gestion de la qualité des Web Services composés par recomposition dynamique. Cette recomposition consiste à modifier les partenaires du business process qui définit un Web Service composé. Elle s'opère afin d'assurer le respect de contraintes de qualité exprimées à l'aide de l'extension de BPEL4WS pour la qualité de service. Le fonctionnement de la plateforme se décompose entre 5 opérations : **recherche de services** à l'aide d'un annuaire privé (annuaire étendu pour la description non fonctionnelle), **classement des compositions de services potentielles** selon les contraintes de qualité, **monitorage de la composition courante** (avertit si les contraintes de qualité ne sont pas respectées), **gestion** (détermine une nouvelle composition lorsque la composition courante n'est plus satisfaisante) et **routage** (oriente le business process vers les nouveaux partenaires).

Chapitre 5

Prototype de la plateforme

5.1 Introduction

Afin de valider notre modèle, nous avons réalisé une implémentation de notre plateforme sous Java. D'une part, cette implémentation reprend l'architecture en 4 sous-systèmes que nous avons détaillée dans la section précédente. D'autre part, elle utilise un annuaire UDDI privé, une plateforme Axis[12] pour déployer le système de routage (les routeurs) et un réseau WSMN pour monitorer le Web Service composé.

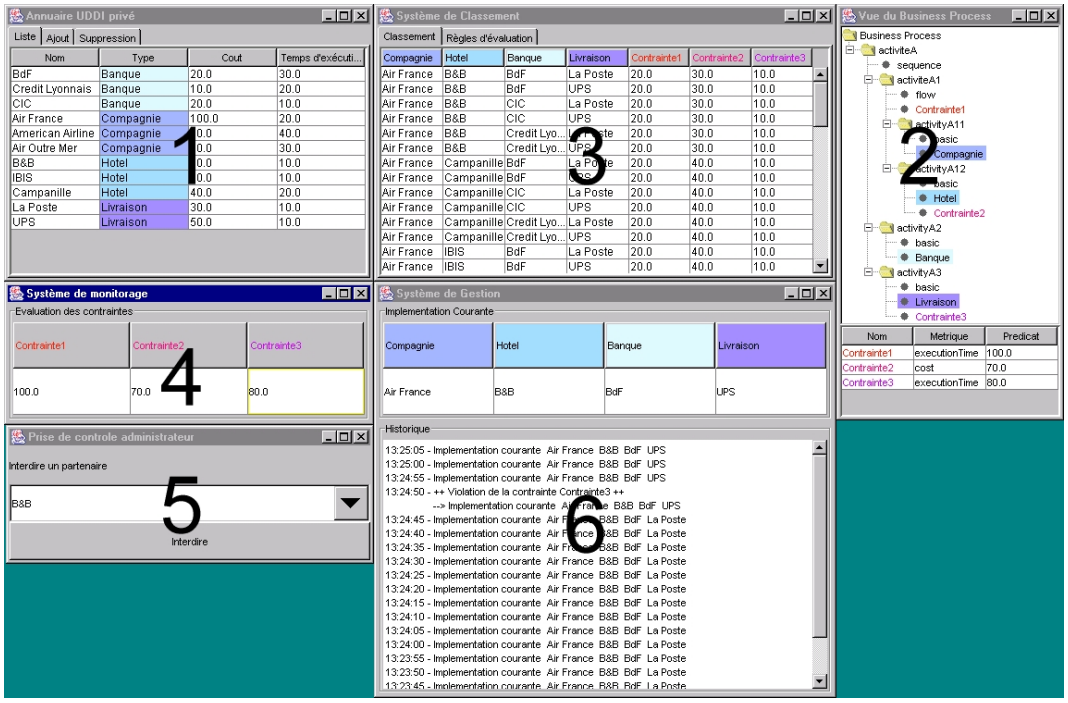


FIG. 5.1 – Capture écran du prototype

[12]Cette plateforme est une implémentation Java de la spécification SOAP. Elle permet de déployer des Web Services.

Sur la figure 5.1, nous pouvons observer les différentes parties du prototype de la plateforme. La fenêtre 1 correspond à l'annuaire UDDI (avec extension). La fenêtre 2 correspond à la vue du business process (muni des contraintes). A partir de l'annuaire et du business process, le système de classement (fenêtre 3) définit et classe les compositions potentielles. Enfin, le système de gestion (fenêtre 6) définit la composition courante en fonction du classement, des alarmes du système de monitorage (fenêtre 4) et des consignes de l'administrateur (fenêtre 5).

Nous allons présenter quelques éléments caractéristiques de l'implémentation de la plateforme.

5.2 Représentation objet du business process

Pour faciliter la manipulation du business process, nous avons implémenté une classe business et une classe activité qui permettent de représenter le business process et son contenu sous la forme objet (figure 5.2).

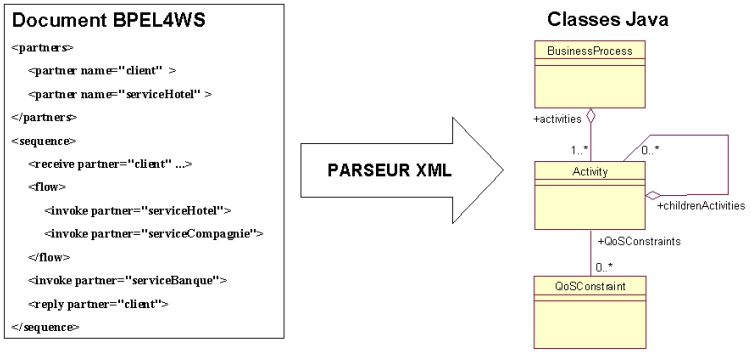


FIG. 5.2 – Modélisation objet du business process

Nous utilisons un parseur XML qui permet de générer automatiquement la version objet du business process à partir du fichier XML BPEL4WS. Cette démarche s'approche des travaux réalisés par IBM Research pour le moteur d'orchestration BPWS4J.

5.3 Système de routage et plateforme Axis

Le système de routage de notre plateforme permet de sélectionner le fournisseur adéquat pour chaque Web Service élémentaire (i.e. pour chaque partenaire) utilisé dans le business process. Un routeur (figure 5.3) est utilisé pour chacun de ces services.

Chaque routeur a été implémenté sous la forme d'un Web Service proxy qui peut réaliser des appels vers les différents fournisseurs correspondant à un même type de Web Services. Les différents routeurs ont été déployés sur la plateforme SOAP Axis.

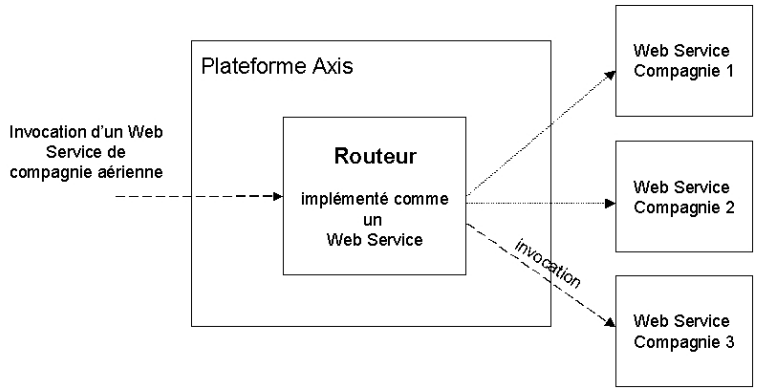


FIG. 5.3 – Routeur sur la plateforme Axis

5.4 Utilisation de l'architecture WSMN pour le monitorage

Le réseau WSMN d'Akhil Sahai permet, à partir d'un business process, d'établir un réseau de proxys pour le monitorage d'un Web Service composé. Les mesures réalisées sont attachées aux différentes branches du business process dans une base de données.
Des appels à cette base permettent de récupérer les mesures. Ces dernières sont par la suite utilisées et traitées pour évaluer si les contraintes de qualité du business process sont respectées.

5.5 Evaluation de la plateforme

Nous avons réalisé plusieurs scénarios d'utilisation de la plateforme, pour tester la recomposition. Nous avons traité les cas suivants :

- **apparition d'un nouveau prestataire plus compétitif** : le classement fait apparaître une nouvelle composition plus intéressante qui provoque la recomposition,
- **dégradation de la qualité de service de partenaires** : certains partenaires fournissent une qualité de service qui se dégrade. Les contraintes de qualité ne sont plus respectées et une recomposition est provoquée.
- **recomposition forcée par l'administrateur** : l'administrateur prend la main et interdit la composition courante.

5.6 Synthèse

L'implémentation de la plateforme (pour la recomposition dynamique) a été réalisée en Java et fait appel à un annuaire UDDI privé, une plateforme Axis et une architecture WSMN. La manipulation du business process (et de son extension) a été facilitée par une modélisation objet. Le système de routage a été conçu sous la forme de Web Services et les mesures du réseau WSMN permettent d'évaluer le respect ou non des contraintes de qualité du business process. Cette implémentation a permis de tester notre plateforme et de mettre en évidence l'intérêt de notre approche.

Conclusion générale

Les Web Services facilitent l'intégration et la communication entre applications sur l'Internet en utilisant des standards XML. Ils correspondent à des composants logiciels déployés sur le Web, qui peuvent être combinés pour former des services à valeur ajoutée, par un mécanisme de composition. Ainsi, le langage BPEL4WS permet d'assembler, sous la forme d'un business process, des Web Services élémentaires pour aboutir à un Web Service plus élaboré.

La forte dynamicité de ces services nécessite de créer de nouvelles infrastructures de supervision mieux adaptées. La supervision des Web Services s'étend de l'évaluation de l'état d'un service à sa configuration en passant par la gestion de la qualité.

Ce stage de DEA a mis en évidence une nouvelle méthode permettant d'exploiter les modèles de composition pour la supervision des Web Services, en traitant le cas de la gestion de la qualité dans les Web Services composés.

Nous avons proposé une extension au langage de composition BPEL4WS, qui permet d'exprimer des contraintes de qualité dans le business process et nous avons créé une plateforme de supervision, qui exploite cette extension. Cette plateforme permet la gestion de la qualité des Web Services par recomposition dynamique. Nous avons développé un prototype de cette plateforme qui a confirmé l'intérêt de notre démarche.

Ce projet traduit clairement le potentiel offert par les modèles de composition pour la supervision de Web Services composés. De nouvelles perspectives sont envisageables en matière de gestion du changement et de gestion de la sécurité, toujours en ajoutant de nouvelles propriétés au business process par le biais d'extensions. Ces voies seront davantage explorées à travers la réalisation d'un stage de 3 mois chez IBM Research à Yorktown NY.

Bibliographie

[1] Keller A., Kar G., Ludwig H., and Dan A. Managing dynamic services : A contract based approach to a conceptual architecture. NOMS 2002, IEEE, April 2002.

[2] Keller A. and Ludwig H. Defining and monitoring service level agreements for dynamic e-business. LISA 2002, November 2002.

[3] Keller A. and Ludwig H. The wsla framework : Specifying and monitoring service level agreements for web services. Technical report, IBM Research, March 2003.

[4] Moorsel A. Ten-step survival guide for the emerging business web. Technical report, Hewlett-Packard Labs, July 2002.

[5] Sahai A., Durante A., and Machiraju V. Towards automated sla management for web services. Technical report, Hewlett-Packard Labs, July 2002.

[6] Sahai A., Machiraju V., and Ouyang J. Message tracking in soap-based web services. NOMS 2002, IEEE, April 2002.

[7] Ensel C. and Keller A. Managing application service dependencies with xml and the resource description framework. IM 2001, IEEE Press, May 2001.

[8] Peltz C. Web services orchestration. Hewlett-Packard Company, January 2003.

[9] Blondeau D., Arkin A., Jekeli W., and Pogliani S. Business process modeling language (bpml) specification. Technical report, Business Process Modeling Initiative (BPMI), November 2002.

[10] Box D., Ehnebuske D., Kakivaya G., and Layman A. Simple object access protocol (soap) specification. Technical report, World Wide Web Consortium, W3C Note, May 2000.

[11] Christensen E., Curbera F., Meredith G., and Weerawarana S. Web services description language (wsdl) specification. Technical report, World Wide Web Consortium, W3C Note, March 2001.

[12] Sahai A. et al. Automated sla monitoring for web services. DSOM 2002, IEEE Press, October 2002.

[13] Curbera F., Goland Y., Klein J., Leymann F., and Roller D. Business process execution language for web services. Technical report, BEA and IBM Research and Microsoft, July 2002.

[14] Kadima H. and Monfort V. *Les Web Services : Techniques, Démarches et outils.* DUNOD, March 2003.

[15] Cardoso J., Sheth A., and Miller J. Workflow quality of service. Technical report, LSDIS Lab, University of Georgia, March 2002.

[16] Farrell J.A. and Kreger H. Web services management approaches. *IBM Systems Journal*, November 2002.

[17] Debusmann M. and Keller A. Sla-driven management of distributed systems using the common information model. IM 2003, IEEE Press, March 2003.

[18] Wohed P., van der Aalst W., and Dumas M. Pattern based analysis of bpel4ws. Technical report, Department of Computer and Systems Sciences, Stockholm University, May 2002.

[19] Levy R., G. Pacifici, and M. Spreitzer. Performance management for cluster based web services. IM 2003, IEEE Press, March 2003.

[20] Sturm R., Morris W., and Mary J. *Foundations of Service Level Management.* SAMS Publishing, January 2000.

[21] Fordin S., Arkin A., Jekeli K., and Orchard D. Web service choreography interface (wsci) specification. Technical report, BEA and Intalio and SAP and Sun, 2002.

[22] Bellwood T., Clément L., Ehnebuske D., and Hately A. Universal description, discovery and integration (uddi) specification. Technical report, OASIS Committee, July 2002.

[23] Machiraju V., Sahai A., and van Moorsel A. Web services management network. IM 2003, IEEE Press, March 2003.

[24] Tosic V., Patel K., and Pagurek B. Web service offerings language. Workshop on Web Services, e-Business and the Semantic Web - WES, CAISE 2002, May 2002.

[25] Diao Y., Eskesen F.N., Froehlich S.E., Hellerstein J.L., and Keller A. Generic on-line discovery of quantitative models for service level management. IM 2003, IEEE Press, March 2003.

Glossaire

Apache Axis : Implémentation Java de la spécification SOAP. Intégrée à un serveur d'applications, cette plateforme permet le déploiement de Web Services.

API (Application Programming Interface) : Une API est une interface de programmation d'application, un jeu de fonctions ou de méthodes, utilisé pour accéder à certaines fonctionnalités.

BPEL4WS (Business Process Execution Language for Web Service) : Langage XML proposé par IBM, Microsoft et BEA pour la spécification formelle de la composition de services dans le cadre des Web Services. Ce langage décrit la composition sous la forme d'un business process abstrait ou exécutable.

Business Process : Ensemble d'activités (utilisant personnes, information et autres ressources) à effectuer pour mener à bien une tâche.

CCM (Corba Component Model) : Modèle de composants multi-langage proposé par l'OMG dans la version 3 de Corba.

Composition de services : Mécanisme qui consiste à assembler des services pour aboutir à un service plus élaboré.

Contrat de service : Le contrat de service est le point clé de la négociation entre un client et un fournisseur de services. Ce document fixe le niveau de service, la disponibilité et les obligations des différentes parties prenantes lors de la négociation.

Corba (Common Object Request Broker Architecture : Spécification définie par l'OMG qui normalise les ORB afin de garantir l'inter-opérabilité des applications conformes à Corba, indépendamment du langage et du système d'exploitation utilisés.

Grid computing : Modèle de système d'information où les ressources sont totalement éclatées à travers un réseau. Ces réseaux-grilles s'appuient sur des protocoles standards et des technologies ouvertes pour associer entre elles des grappes de serveurs tout autour de la planète. L'objectif est de permettre à des organisations dispersées, formelles ou informelles, de partager des applications, des données et des ressources comme de la puissance de calcul ou de l'espace disque.

HTTP (HyperText Transport Protocol) : Protocole qui permet au client du Web de recevoir le contenu de documents hypertextes (format HTML).

Java : Langage de programmation orienté objet multi plates-formes développé par la société Sun Microsystems.

JMX (Java Management Extension) : Standard destiné à l'administration et la supervision d'applications Java à travers le réseau. JMX définit une couche d'isolation entre les ressources à gérer (objets Java) et le système d'administration.

Métrique : Mesure ou variable permettant de quantifier un aspect d'un service ou d'un réseau (exemples : temps de réponse, temps d'exécution, coût).

MOM (Message Oriented Middleware) : Middleware orienté message, mécanisme de communication par message asynchrone, les MOM offrent des mécanismes adaptés pour

connecter des applications distribuées.

Monitorage : Ensemble des techniques permettant d'analyser et d'évaluer l'état d'un service ou d'un élément réseau.

ORB (Object Request Broker) : Logiciel offrant des services de communications normalisées permettant à des objets répartis d'inter-agir.

Plateforme de services : Structure permettant la conception, la mise en œuvre et le déploiement de services.

Qualité de service (QdS ou QoS) : Ensemble de critères souvent contractuels permettant de définir le niveau de performance attendu d'une application, d'un service, d'un réseau ou de tout autre dispositif. Par exemple pour un réseau, les critères utilisés sont en général la bande passante, le temps de latence, la perte de paquets et le temps de réponse à une requête.

Routage : Méthode d'acheminement des informations à la bonne destination à travers un réseau.

SLA (Service Level Agreement) : Voir contrat de service.

SMTP (Simple Mail Transfer Protocol) : Protocole de base qui permet aux serveurs de messagerie de l'Internet de communiquer.

SOAP (Simple Object Access Protocol) : Protocole visé par W3C pour l'échange d'informations en environnement distribué et décentralisé. SOAP est intégralement basé sur XML. Il spécifie le format des messages échangés, les règles de codage et les conventions de passage de paramètres. Il permet l'appel de méthodes sur des objets distants en utilisant XML. SOAP est le plus souvent utilisé avec HTTP comme protocole de transport.

Supervision : Activité qui consiste à surveiller et contrôler un service ou élément réseau pour qu'il satisfasse les besoins des clients et les contraintes des fournisseurs.

UDDI (Universal Description, Discovery, and Integration) : Standard proposé par OASIS permettant la description, l'enregistrement et la recherche d'informations relatives à un Web Service. UDDI définit à la fois un protocole basé sur XML pour la publication et la recherche de Web Services et à la fois un modèle d'annuaire pour les Web Services. Un annuaire UDDI contient des informations telles que les intervenants du e-business (les sociétés, administrations) et les méta-données des Web Services proposés (moyens d'accès, paramètres et valeurs de retour). UDDI est une réalisation conjointe d'IBM, Microsoft et Ariba.

URL (Uniform Ressource Location : Adresse qui spécifie la localisation physique d'un répertoire, d'un fichier ou plus généralement d'une ressource se trouvant sur le World Wide Web.

Web : Le World Wide Web (souvent abrégé Web ou WWW) est un service de l'Internet qui utilise le protocole HTTP (Hyper Text Transport Protocol) et aussi le protocole FTP dans une moindre mesure. Le Web permet aux entreprises et aux particuliers de créer des serveurs d'informations sur l'Internet.

Web Services : Composants logiciels faiblement couplés qui s'exécutent au travers de l'infrastructure Web et qui peuvent être décrites, publiées auprès d'un annuaire et invoquées par des applications clientes. Ce modèle repose sur les trois standards SOAP, WSDL et UDDI.

Web Service composé : Web Service issu d'une composition.

WSDL (Web Service Description Language) : Format de description de Web Service. Ce standard visé par W3C permet de définir les méta-données associées à un Web Service. Ces méta-données décrivent les types de données manipulées, les mécanismes d'appels et les opérations supportées.

WSLA (Web Service Level Agreement) : Langage XML proposé par IBM Research pour l'établissement de contrats de service dans le cadre des Web Services.

WSLA framework : Plateforme proposée par IBM Research pour l'établissement et le contrôle de contrats de service dans le cadre des Web Services.

WSMN (Web Service Management Network) : Réseau d'overlay au-dessus de SOAP pour le monitorage et la gestion de fautes des Web Services composés.

XML (eXtensible Markup Language) : Format universel de stockage et d'échange de données. XML est un langage de balisage extensible, c'est-à-dire qu'il n'est pas sémantiquement figé comme HTML. Il permet de définir ses propres balises, ce qui le rend adaptable et donc à même de stocker tous types d'informations.

XML Schema : Langage XML de description de types de documents XML.

Annexes

Annexe A

Spécification de l'extension de BPEL4WS pour la qualité de service

XML Schema de l'extension

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

   <redefine schemaLocation="ws/2003/03/business-process/bpel4ws.xsd">
      <complexType name="tActivity">
         <annotation>
            <documentation>Redefinition du type activité pour l'ajout d'un attribut
            QoSConstraints qui contient les contraintes de qualité associées à
            l'activité.</documentation>
         </annotation>
         <complexContent>
            <extension base="bpws:tExtensibleElements">
               <sequence>
                  <element name="target" type="bpws:tTarget"
                           minOccurs="0" maxOccurs="unbounded"/>
                  <element name="source" type="bpws:tSource"
                           minOccurs="0" maxOccurs="unbounded"/>
               </sequence>
               <attribute name="name" type="NCName"/>
               <attribute name="joinCondition" type="bpws:tBoolean-expr"/>
               <attribute name="suppressJoinFailure" type="bpws:tBoolean" default="no"/>
               <attribute name="QoSConstraints" type="IDREFS"/>
             </extension>
         </complexContent>
      </complexType>
   </redefine>

   <element name="QoSConstraints">
      <annotation>
         <documentation>Definition de l'ensemble des contraintes de qualité du business
         process</documentation>
```

```
      </annotation>
      <complexType>
         <sequence minOccurs="0" maxOccurs="unbounded">
            <element name="QoSConstraint"/>
         </sequence>
      </complexType>
   </element>

   <element name="QoSConstraint">
      <annotation>
         <documentation>Définition d'une contrainte de qualité</documentation>
      </annotation>
      <complexType>

         <attribute name="name" type="ID" use="required">
            <annotation>
               <documentation>L'attribut name correspond au nom de la contrainte.
               Ce nom est référencé par une activité.</documentation>
            </annotation>
         </attribute>

         <attribute name="type" use="required">
            <annotation>
               <documentation>L'attribut type définit la portée d'une
               contrainte.</documentation>
            </annotation>
            <simpleType>
               <restriction base="string">
                  <enumeration value="basic"/>
                  <enumeration value="structured"/>
                  <enumeration value="inter"/>
               </restriction>
            </simpleType>
         </attribute>

         <attribute name="metric" type="string" use="required">
            <annotation>
               <documentation>L'attribut metric correspond à la métrique sur laquelle porte
               la contrainte.</documentation>
            </annotation>
         </attribute>

         <attribute name="metricType" type="anyURI" use="optional">
            <annotation>
               <documentation>L'attribut metricType permet de référencer un type
               de métrique dans un fichier de description WSOL ou WSEL.</documentation>
            </annotation>
         </attribute>
   </element>

</schema>
```

Annexe B

Exemple d'utilisation de l'extension

Business process du service de réservation

```
<process xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/"
         name="processusReservationAgenceDeVoyages"
         targetNamespace="http://tempuri.org/"
         xmlns:tns="http://tempuri.org/"
         suppressJoinFailure="no"
         containerAccessSerializable="no"
         enableInstanceCompensation="no"
         abstractProcess="no">

<QoSConstraints>

   <QoSConstraint name="contrainteA"  type="structured"
                  metric="executionType" predicat="inf100" />

   <QoSConstraint name="contrainteB"  type="basic"
                  metric="cost" predicat="inf1000;sup100" />

   <QoSConstraint name="contrainteC"  type="basic"
                  metricType="http://www.banque.com/descriptionQoS.wsel"
                  metric="responseTime" predicat="inf10" />

</QoSConstraints>

<partners>

   <partner name="client"
      partnerLinkType="reservationSejourLT"
      partnerRole="clientAgenceDeVoyages"
      myRole="agenceDeVoyages"/>

   <partner name="serviceCompagnie"
      partnerLinkType="reservationBilletsAvionLT"
      partnerRole="fournisseurBilletsAvion"
      myRole="clientBilletsAvion"/>

   <partner name="serviceHotel"
      partnerLinkType="reservationChambreLT"
```

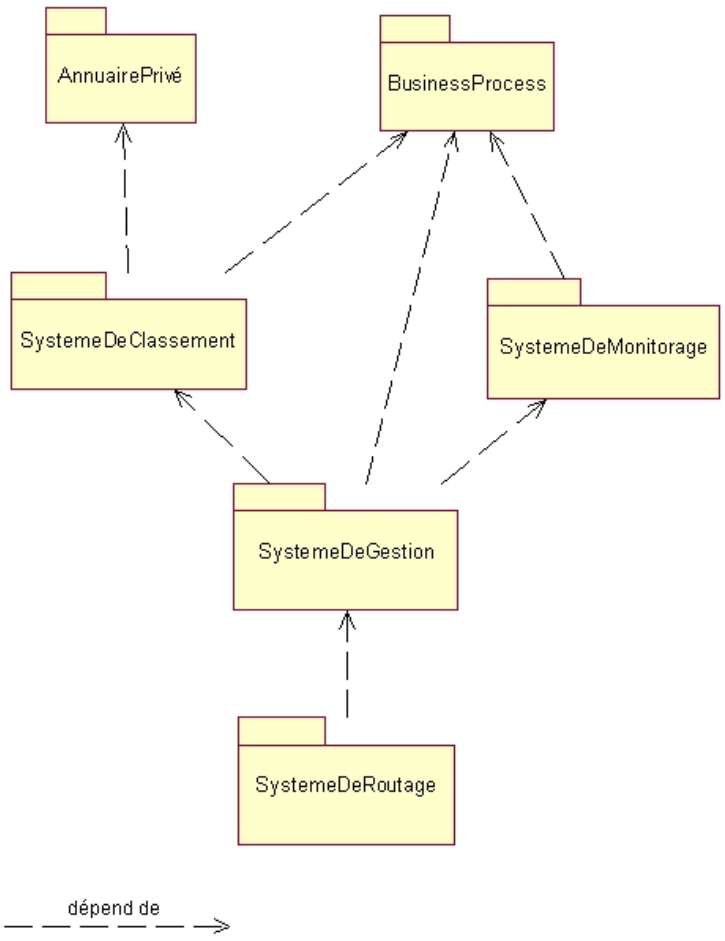
```
      partnerRole="fournisseurReservationChambre"
      myRole="clientReservationChambre"/>

   <partner name="serviceBanque"
      partnerLinkType="prelevementLT"
      partnerRole="fournisseurPrelevement"
      myRole="clientPrelevement"/>

   <partner name="serviceLivraison"
      partnerLinkType="envoiFactureLT"
      partnerRole="fournisseurEnvoiFacture"
      myRole="clientEnvoiFacture"/>

</partners>

<variables>

   <variable name="demande" messageType="typeCommande"/>
   <variable name="recepisse" messageType="typeRecepisse"/>
   <variable name="demandeBillets" messageType="typeDemandeBillets">
   <variable name="recepisseBillets" messageType="typeRecepisseBillets">
   [...]

</variables>

<sequence>

   <receive partner="client"
            portType="reservationSejourPT"
            operation="reservationSejour"
            variable="commande">
   [...]

   <flow QoSConstraints="contrainteA">

       <invoke partner="serviceCompagnie" QoSConstraints="contrainteB"
       portType="reservationBilletsPT"
       operation="reservationBillets"
       input="demandeBillets"
       output="recepisseBillets">

       <invoke partner="serviceHotel"
               portType="reservationChambrePT"
               operation="reservationChambre"
               input="demandeChambre"
               output="recepisseChambre">
   </flow>

   <invoke partner="serviceBanque" QoSConstraints="contrainteC"
           portType="prelevementPT"
           operation="prelevement"
           input="demandePrelevement"
           output="recepissePrelevement">
   [...]
```

```
   <reply partner="client"
          portType="reservationSejour"
          operation="reservationSejour"
          variable="recepisse">

   <invoke partner="serviceLivraison"
           portType="envoiFacturePT"
           operation="envoiFacture"
           input="demandeEnvoiFacture"
           output="recepisseEnvoiFacture">

</sequence>

</process>
```

Annexe C

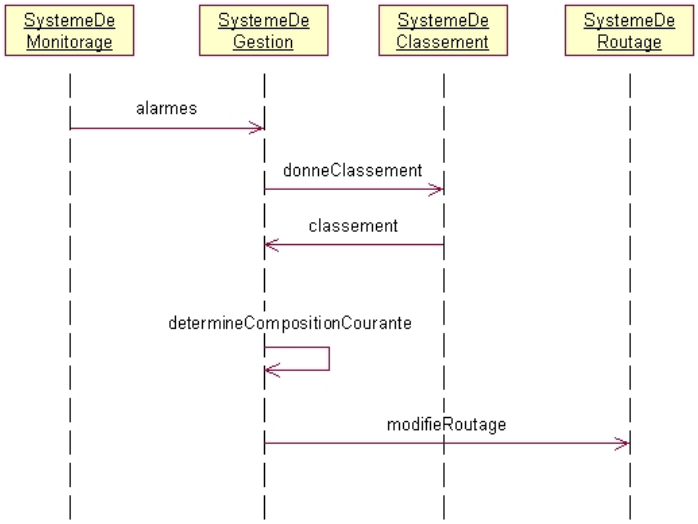
Diagrammes de la plateforme de gestion de la qualité par recomposition

Diagramme des dépendances

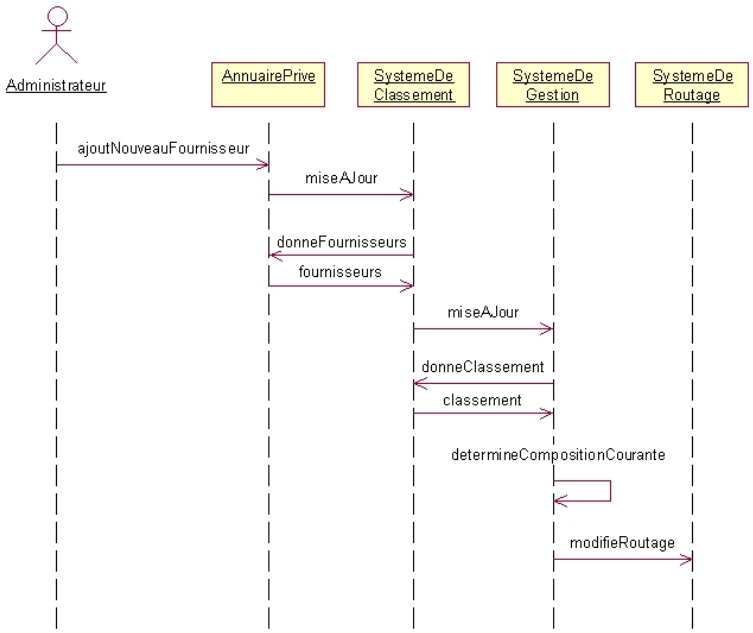


Diagrammes de séquence

Recomposition suite à une alarme du système de monitorage



Recomposition suite à l'apparition d'un nouveau fournisseur compétitif



Recomposition dictée par l'administrateur